# Data Structures Exam: Olympics

## Description

You have to implement a structure that keeps track of the Olympic scores in Tokyo 2020. Your structure will have to support the following functionalities:

* **AddCompetitor(competitorName, competitorId)** – you have to register a new competitor. If there is already a competitor with that id, return ArgumentException
* **AddCompetition(compoetitionName, competitionId, score)** – you have to register a new competition. If there is already a competition with that id, return ArgumentException
* **Compete(competitorId, competitionId, score)** – search for the given competitor and the given competition and add the competitor with his score to the given competition.   
  **If there are no such competitor or competition throw ArgumentException**
* **Disqualify(competitionId, competitorId)** – search for the given **competitor** in the given **competition** and **remove him from the competition**. **Reduc**e the competitor's **total points** by the **amount he lost by being disqualified**. – вадиме от състезателя точките на състезанието  
  **If there are no such competitor or competition throw ArgumentException**
* **GetByName(name)** – return all competitors with the provided name ordered by their id’s. If there is no competitor with the given name return ArgumentException
* **FindCompetitorsInRange(start, end)** – find all competitors which have total score between the given start exclusive and end inclusive.
* SearchWithNameSize(int, int) – Returns all competitions that have name lengths between the 2 parameters inclusive and order them by id. If there aren’t any return empty collection.
* Contains(competitionId, Competitor) – Checks if a competitor is present in the competition.
* GetCompetition(id) – return the competition with the given id. If there is no such throw AgumentException
* CompetitorsCount() – return count of registered competitors
* CompetitionsCount() – return count of competitions

*Feel free to override Equals() and GetHashCode() if necessary.*

## Input/Output

You are given a **Visual Studio C# project skeleton** (unfinished project) / **IntelliJ Java project** holding the interface IOlympics, the unfinished classes Competition, Competitor and Olympics. **Tests** covering the Olympics **functionality** and **performance**.

Your task is to **finish this class** to make the tests run correctly.

* You are **not allowed to change the tests**.
* You are **not allowed to change the interface**.
* You can add to the Olympics class, but don't remove anything.
* You can edit the Olympics class if it implements the IOlympics interface.

## Interface

The interface IOlympics in C# looks like the code below:

|  |
| --- |
| public interface IOlympics  {  void AddCompetitor(int id, string name);  void AddCompetition(int id, string name, int score);  void Compete(int competitorId, int competitionId);  void Disqualify(int competitionId, int competitorId);  IEnumerable<Competitor> FindCompetitorsInRange(long min, long max);  IEnumerable<Competitor> GetByName(string name);  IEnumerable<Competitor> SearchWithNameLength(int min, int max);  bool Contains(int competitionId, Competitor comp);  int CompetitionsCount();  int CompetitorsCount();  Competition GetCompetition(int id);  } |

The interface Olympics in Java looks like the code below:

|  |
| --- |
| **public interface Olympics {**  **void addCompetitor(int id, String name);**  **void addCompetition(int id, String name, int score);**  **void compete(int competitorId, int competitionId);**  **void disqualify(int competitionId, int competitorId);**  **Iterable<Competitor> findCompetitorsInRange(long min, long max);**  **Iterable<Competitor> getByName(String name);**  **Iterable<Competitor> searchWithNameLength(int minLength, int maxLength);**  **Boolean contains(int competitionId, Competitor comp);**  **int competitionsCount();**  **int competitorsCount();**  **Competition getCompetition(int id);**  **}** |

## Submission

Submit an archive (.zip) of the source code. Your code **mustn't** contain namespaces/packages.

## Scoring

Each implemented method brings you a specific amount of points, some of the points are awarded for correct behavior, others for performance. **The performance tests might not work on your PC**. You need to cover all tests in each group to receive points. Bellow is a breakdown of all points by methods:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Correct Behavior | Performance | Total |
| Overall | 50 | 100 | 150 |